Microsoft
.net developer
training
Create XML Web Services and Applications with Visual Studio .NET

Building Applications for Mobile Devices

Agenda: Day 2

Module 09: Creating XML Web Services

Module 10: Code Instrumentation and Middle Tier Technology

Module 11: Security in .NET

Module 12: Lifecycle Tools for .NET

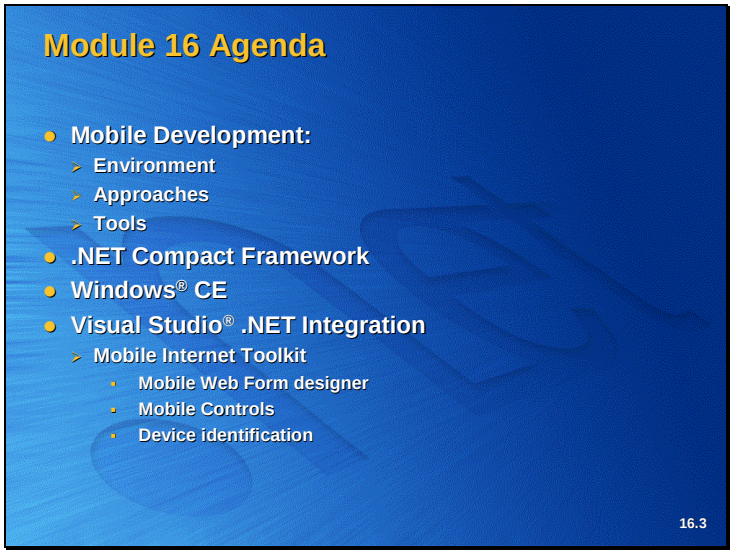
Module 13: Configuring and Deploying .NET Applications

Module 14: Application Interoperability

Module 15: Migrating Applications to .NET

▶ Module 16: Building Applications for Mobile Devices

 ©2002 Microsoft Corporation



Module 16 Agenda

What you will learn:

How Visual Studio enables you to target mobile devices via a browser and via client-side code:

- ***Via a browser***: How to write Web applications in Visual Studio .NET that can talk to your Mobile devices
- ***Via client-side code***: How Visual Studio .NET and the Mobile Internet Toolkit automatically create the right client code for each device

Mobile Development

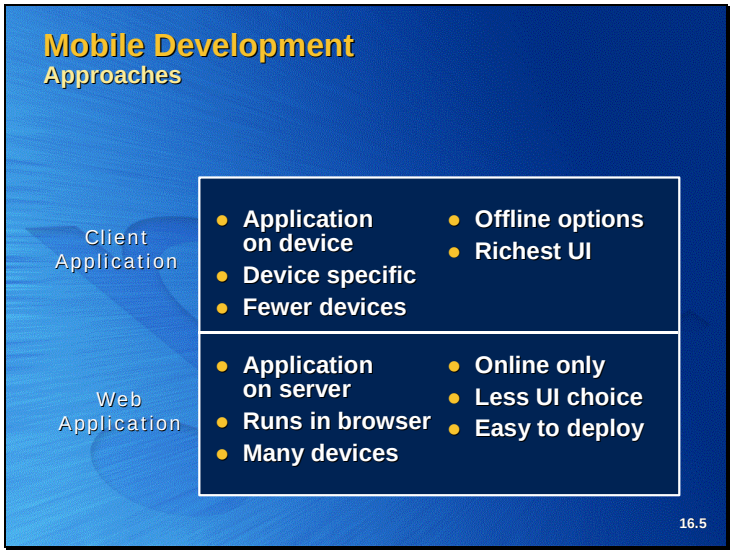
Environment

Visual Studio .NET is the code development, debugging and code management environment for development to all of these technologies. Using Visual Studio .NET, you can build the server-side business logic to drive your applications, rich client applications that run on your mobile devices such as the Pocket PC, Mobile Web Applications that run in mobile browsers of all sorts and XML Web services that publish information to mobile devices.

You also can write applications that consume these XML Web services on the mobile devices.

Visual Studio .NET ties everything together.

 ©2002 Microsoft Corporation



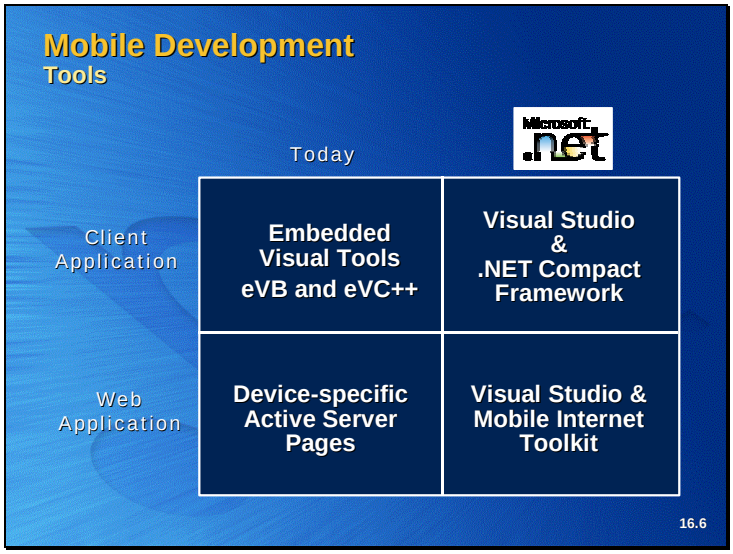
Mobile Development

Approaches

Client-side applications run on the device while Mobile Web Applications run on the server. Both have their purpose so the development scenario determines which is most appropriate.

Client-side applications can readily take advantage of device-specific capabilities, but it is harder to make them run on a broad range of different devices. Client-side applications can run offline, they have the richest UI possibilities, and most importantly, they have a powerful local data store.

In contrast, Mobile Web Applications are primarily on the server, but render in the mobile browser. It is easier to target a wide range of devices with Mobile Web Applications. Disadvantages include the requirement of connectivity to a Web server, frequent round trips to the server, differences between browsers (i.e. Script support vs. non script support) and limits to the UI capabilities of the client browser. However, they are easy to deploy because no setup is required.



Mobile Development

Tools

Visual Studio client side desktop applications are built by coding against the .NET Framework for the desktop. Applications created for devices will be coded against the .NET Compact Framework (a subset of the desktop framework).

In the past, developers could build client-side applications for devices running Windows CE by using the embedded Visual Tools that consist of eVB (embedded Visual Basic) and eVC (embedded Visual C++). Going forward, developers will be able to build client-side applications for the PocketPC using Visual Basic .NET and C# - directly from within Visual Studio .NET! No longer will there be one tool for desktop development and another for device development. If you can program for the desktop, you can program for a device, it is that simple. Additional languages will even come on later!

By integrating device-programming capabilities directly into Visual Studio, developers can use the same designers, debugger, code editor, control anchoring and docking, and virtually everything else that Visual Studio .NET provides for developing desktop applications. The benefits of being able to develop for mobile devices extend even deeper than the great visual designers. The client-side applications that you create will be taking advantage of the state of the art .NET Compact Framework itself. eVB today is an interpreted, script-based development environment. Visual Basic .NET applications that target devices utilize the same great .NET technology as the desktop, including the Just-In-Time compiler (so your PocketPC applications will benefit from the speed of native code), modern object oriented techniques, strongly typed code, threading, and networking support . Not only that, no longer will you have to recompile your application for each PocketPC device. When you write applications for the .NET Compact Framework, you simply compile your application once and it will run on all PocketPC devices.

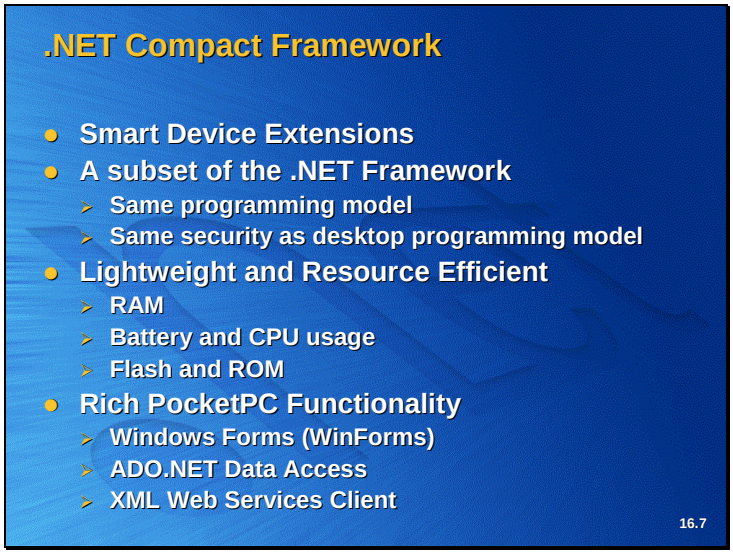
Visual Studio mobile device support will include client-side development for the PocketPC (which is based on Windows CE 3.0) as well as future devices built on Windows CE .NET (also known as Windows CE 4.0, and ‘Talisker’). Note: device programming capabilities (the ‘Smart Device Extensions) will become available for Visual Studio .NET after Visual Studio .NET ships. This is made possible due to the modular, plug and play nature of the Visual Studio IDE itself (the same way that languages and tools insert

 ©2002 Microsoft Corporation

themselves into Visual Studio .NET today). Ship dates for these additional features have not yet been announced.
If I were writing a pre- **.NET Mobile Web Application**, I would build device-specific Active Server Pages, hand-coding for WML for certain phones, or HTML for certain PDAs.

With **.NET -**As an application developer, you can design and develop browser-based applications by using the integrated developer environment (IDE) provided by the Microsoft Mobile Web Form Designer in Visual Studio .NET or you can use command line tools with the Microsoft .NET Framework tools and the Common Language Runtime.

All of today's device development options remain open to the developer, although the new .NET technology offers two new approaches. If you are developing rich client-side applications, you can use the Visual Studio .NET and the .NET Compact Framework, but when building (browser-based) Mobile Web Applications, the integrated Mobile Internet Toolkit will simplify the process.



.NET Compact Framework

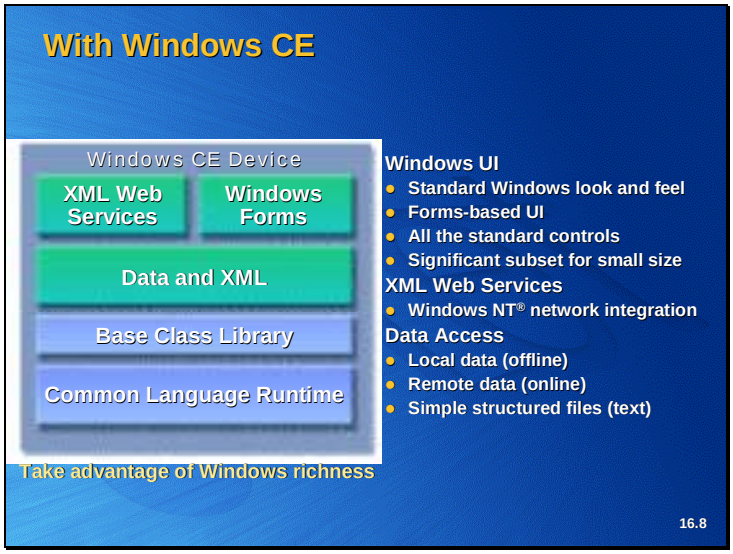
Later in 2002, Microsoft will ship the Smart Device Extensions (SDE) for Visual Studio .NET. SDE, when installed into Visual Studio .NET, enables the Visual Studio .NET developer to easily create client-side applications for the Pocket PC, the Pocket PC 2002, and devices based on the new Windows CE .NET operating system. SDE consists of the .NET Compact Framework as well some device-specific features, including superior device emulation technology, remote device debugging, and more.

The .NET Compact Framework is a subset of the .NET Framework. It is specifically designed for smart, resource-constrained devices. It contains the same CLR and JIT architecture but minimizes the resource requirements on **RAM, battery and CPU usage** and adds support for both **Flash memory and ROM.**

Like the desktop .NET framework, the .NET Compact framework will provide a rich development platform including powerful class libraries for forms, graphics and controls, **ADO.NET data access** with support for disconnected datasets, and even the ability to create applications that easily consume Web services.

The most important thing to remember about creating applications for devices using Visual Studio .NET is that the process is nearly identical to the techniques that you have learned throughout the .NET Readiness training. If you can program for the desktop, it is a given that you can program for the PocketPC and beyond. PocketPC will simply be an additional “project type” in the File New dialog. On the desktop, you create a form, draw some controls, add some code, press F5 and the application runs. When you are creating a program for a device, the process is the same. However, when you press F5, you will either see the application run on screen within a PocketPC emulator or the application will automatically get downloaded to your device.

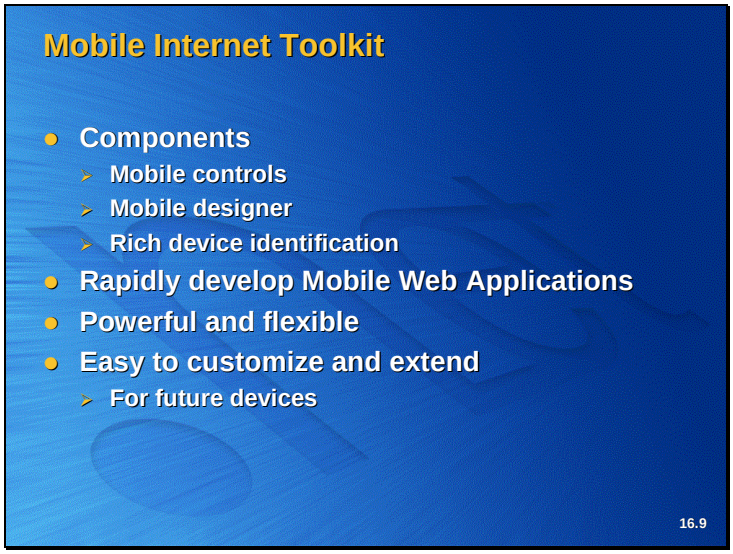
 ©2002 Microsoft Corporation



With Windows CE

Just like the desktop .NET framework, the .NET Compact Framework has three important components:

- A layer that can consume XML Web services running on other machines.
    - Frameworks for **data access** including working with local data, remote data and structured files.
- A base class library, which is a subset of the full-featured .NET Framework.
- The Common Language Runtime, which is a version of the .NET Framework's CLR adapted to run efficiently on mobile devices.



Mobile Internet Toolkit

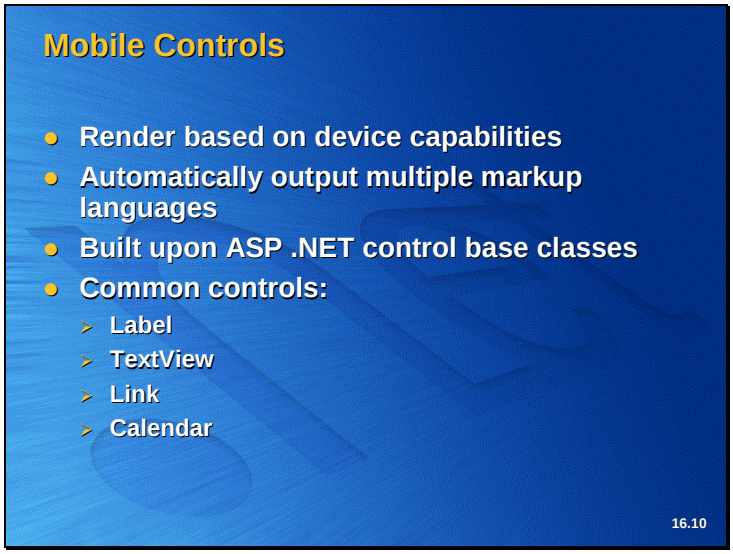
Just as the .NET Compact Framework is the .NET technology for building client-side applications, the **Mobile Internet Toolkit** is the .NET technology for building Mobile Web Applications, applications that live on the server, but run in the mobile browser.

The Mobile Internet Toolkit consists of three components. There is a set of server-side **mobile controls**, a **Mobile Web Form Designer** that allows you to build a Mobile Web Application with drag and drop abilities as easy as putting together a Visual Basic application. There is also a **rich device identification** mechanism that provides the controls and Mobile Web Applications a consistent device schema.

The Mobile Internet Toolkit includes everything necessary to build Mobile Web Applications quickly. Using the Visual Studio .NET Designer, you can drag and drop mobile controls onto a form and connect them to application logic with C#, Visual Basic .NET or any of the other languages that run on the CLR. The technology is built on top of ASP .NET so the application model is powerful and flexible. You can reuse your existing Web development expertise to build Mobile Web Applications.

There is a strong extensibility model that makes support able to be readily added for new devices without changes to the application. Yet, applications are not relegated to using the lowest-common denominator. Through ASP .NET's templating mechanism, we can build sophisticated UIs.

 ©2002 Microsoft Corporation

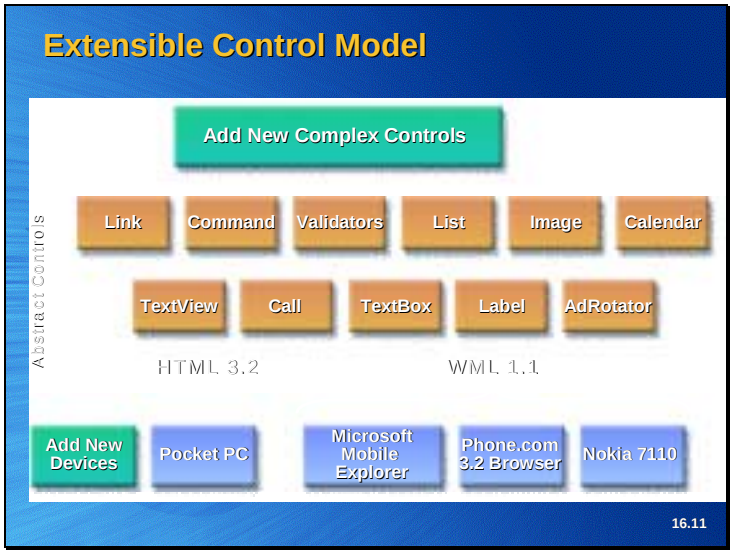


Mobile Controls

The first component of the Mobile Internet Toolkit is a set of **mobile controls** that render according to the capabilities of the device. For example, there is a list control and it knows to paginate differently on a four-line display than on a 10-line display. Since there are numerous markup languages in mobile browsers, the controls know to render HTML to PocketPCs, WML to WAP phones and CHTML to iMode phones. Furthermore, since WML phones differ, the controls can output different markup for different WML devices.

To enable the controls to work with the same ASP .NET application model used for the desktop Web, we build the mobile controls upon the ASP .NET base classes. If you know how to build a desktop Web application with ASP .NET, it is trivial for you to learn how to build a mobile Web application.

The mobile control classes mirror the controls that you get in mobile browsers and are similar to the ASP .NET Web Forms controls. Examples of our common controls include labels, TextViews, links and calendar controls.

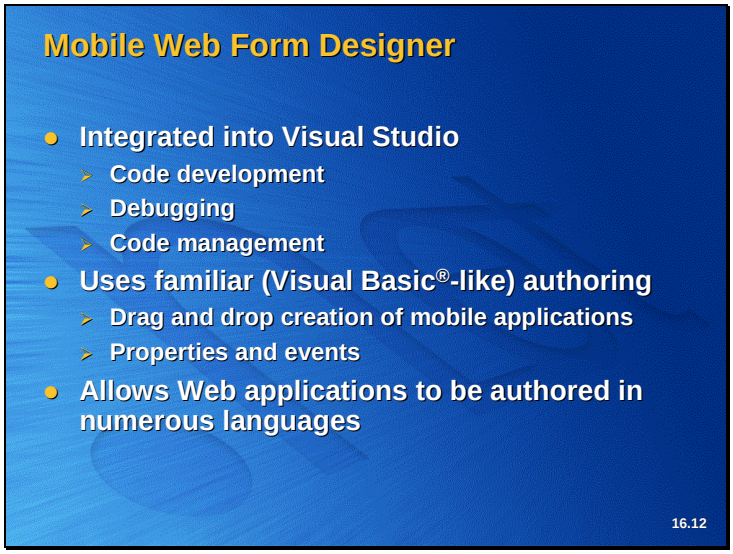


Extensible Control Model

Now, the control model is extensible in three ways.

1. You can build new mobile controls by extending mobile control classes.
2. You can build new complex controls by aggregating several abstract controls.
3. You can add support for new devices through another extensibility mechanism. This support can be added even after an application has been written and deployed.

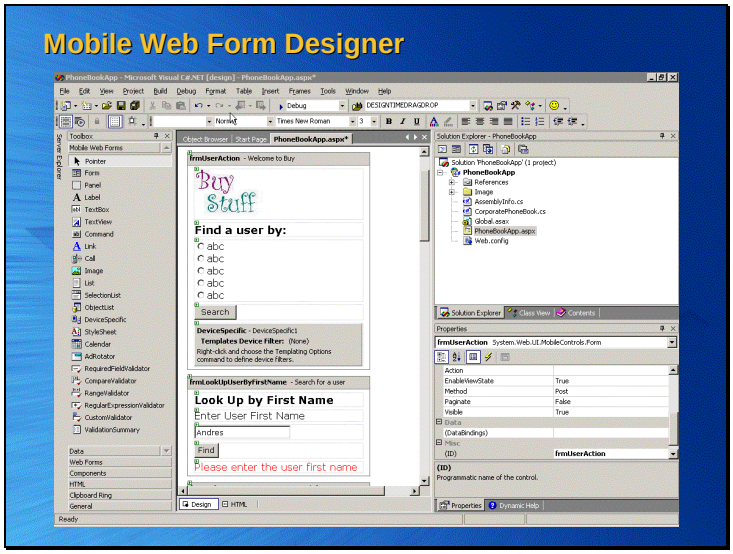
 ©2002 Microsoft Corporation



Mobile Web Form Designer

The second part of the Mobile Internet Toolkit is the **mobile designer** integrated into Visual Studio so you get the same **code development, debugging and code management** environment that you get for building desktop Web applications.

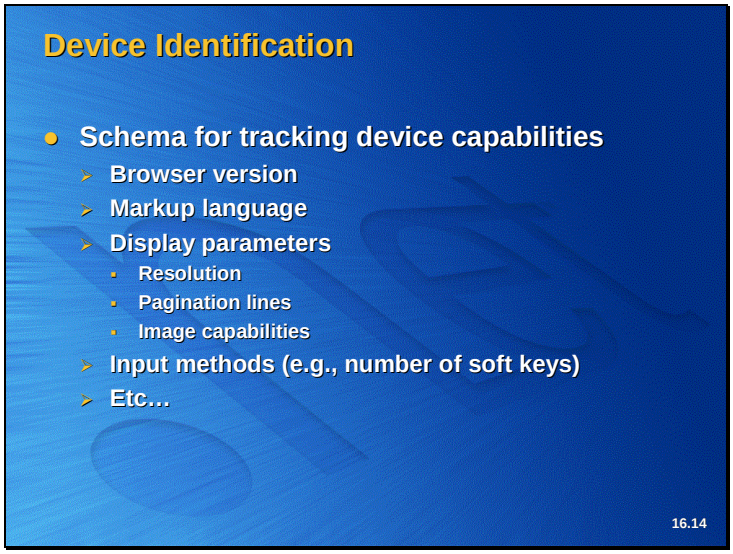
Building Mobile Web Applications is similar to building Visual Basic applications. You can drag and drop mobile controls onto a form and set properties and events, but you need not be authoring in Visual Basic. You can author your application in C#, Visual Basic .NET, Jscript, Perl, Eiffel, Scheme or any of the other languages that run on the Common Language Runtime.



Mobile Web Form Designer

Here is a screenshot of the Mobile Designer in Visual Studio .NET. If you have built an ASP .NET Web Forms application using beta versions of Visual Studio .NET, this should look familiar.

 ©2002 Microsoft Corporation

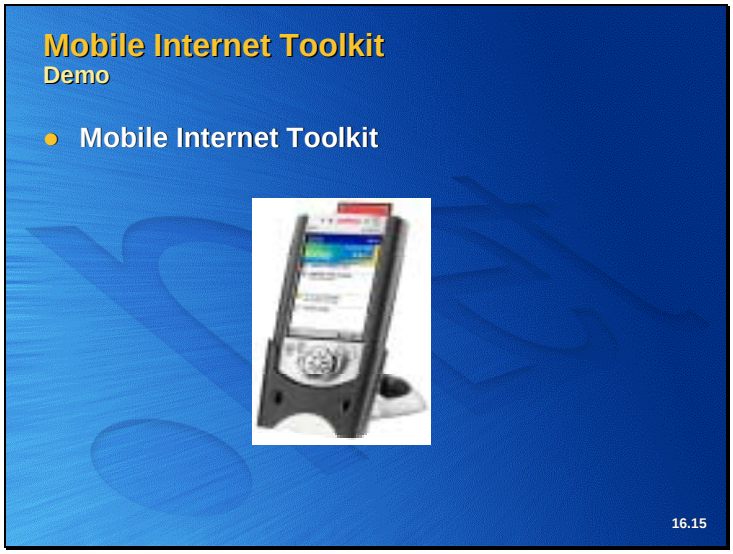


Device Identification

The third component of the Mobile Internet Toolkit is a rich device identification component that includes code for identifying what device is making a request to the server. It provides the controls as well as the application with a consistent schema for representing the capabilities of the device.

For example, the schema includes information on the browser version and the markup language that the browser supports. It contains information on the display – for example, the physical resolution, color depth, numbers of lines of text to display before paginating the content and which image formats the browser supports.

The schema also includes information on the device input methods. For example, mobile browsers often differ in the number of soft keys made available to the browser.



Mobile Internet Toolkit

Demo

The purpose of this demo is to illustrate the Mobile Internet Toolkit.

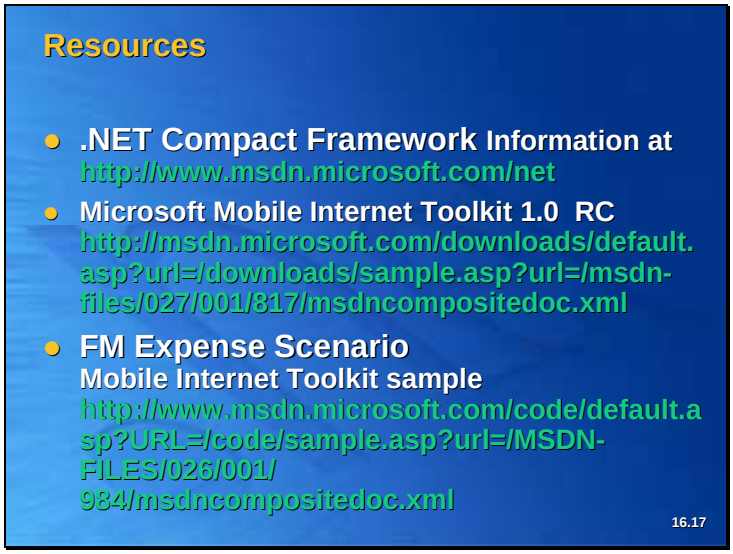
 ©2002 Microsoft Corporation

Summary
For client-side development, there is the .NET Compact Framework
For mobile Web application development, there is the Mobile Internet Toolkit
Both make it easy to develop mobile applications, but different kinds of mobile applications; choose the right tool for your application
16.16

Summary

In summary, there are a number of exciting technologies that are bringing mobile Web development into the .NET age. For client-side development, there is the Smart Device Extensions and the .NET Compact Framework and for mobile Web application development, there is the Mobile Internet Toolkit.

Both make it easy to develop mobile applications, but different kinds of mobile applications; so choose the right tool for the right application.



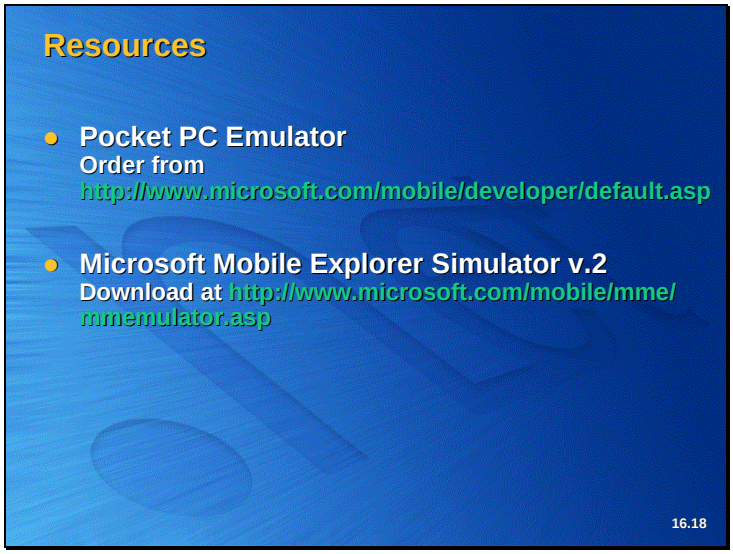
Resources

.NET Compact Framework Information at:
http://www.msdn.microsoft.com/net

Microsoft Mobile Internet Toolkit 1.0 Release Candidate
http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/817/msdncompositedoc.xml

FM Expense Scenario Download this Mobile Internet Toolkit sample from:
http://www.msdn.microsoft.com/code/default.asp?URL=/code/sample.asp?url=/MSDN-FILES/026/001/984/msdncompositedoc.xml

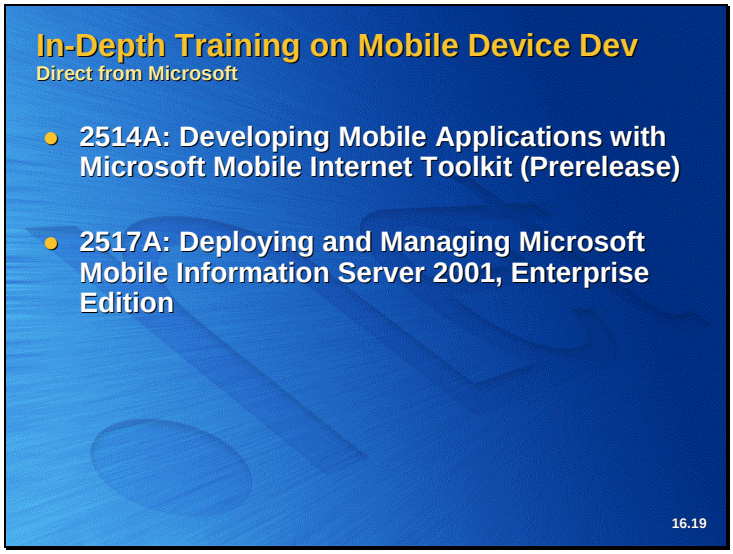
 ©2002 Microsoft Corporation



Resources

Pocket PC Emulator. Order from:
http://www.microsoft.com/mobile/developer/default.asp

Microsoft Mobile Explorer Simulator v.2. Download at:
http://www.microsoft.com/mobile/mme/mmemulator.asp


In-Depth Training on Mobile Device Dev
Direct from Microsoft
2514A: Developing Mobile Applications with Microsoft Mobile Internet Toolkit (Prerelease)
2517A: Deploying and Managing Microsoft Mobile Information Server 2001, Enterprise Edition
16.19

In-Depth Training on Mobile Device Dev

Direct from Microsoft

 ©2002 Microsoft Corporation

Microsoft
16.20